BUFFER MANAGEMENT SIMULATION IN ATM NETWORKS

E. Yaprak, Y. Xiao, and A. Chronopoulos
Wayne State University, Detroit MI 48202

Edward Chow
Jet Propulsion Laboratory, Pasadena, CA 91109

L. Anneberg
Lawrence Technological University, Southfield, MI 48075

ABSTRACT

Computer networking communications are now in the era of dedicated, high-speed switched networks. Asynchronous Transfer Mode (ATM) networks are increasing in importance because ATM technology has the ability to deliver a high quality of service, while simultaneously supporting multiple classes of traffic on the same transmission pathway. Traffic Management is increasingly becoming important focus of study in ATM networks. In this context, congestion control through adequate buffering is becoming particularly significant to minimize the probability of cell loss and cell delay when multiple large traffic bursts are received concurrently at a switch [1,2].

This paper presents a simulation of a new dynamic buffer allocation management scheme in ATM networks. To achieve this objective, an algorithm that detects congestion and updates the dynamic buffer allocation scheme was developed for the OPNET simulation package via the creation of a new ATM module. This dynamic buffer allocation scheme utilizes the four identified attributes: in-use bandwidth, in-use bandwidth of distinct Quality of Service (QoS) of an incoming traffic, available number of buffers on the corresponding logical queue, and the latest short-term cell arrival rate destined to the corresponding output port. The behavior of the network under bursty traffic conditions was examined.

KEY WORDS: ATM Switch, buffer management, congestion control simulation.

1. INTRODUCTION

Current technological advances and significant developments in the computer networking industry are positively influencing the world community daily and helping the industry migrate to next generation technology. Among the many factors that influence these changes, the emergence of high performance computing platforms for graphics, color, voice, video and multimedia applications and the exponential growth in data traffic rates appear to be particularly important. This is partly because these changes are creating an interesting problem for traditional data networks. In shared-medium LAN technologies such as Ethernet, Token Ring, and Fiber Distributed Data Interface (FDDI), for example, only a portion of the total network bandwidth is provided for each user. Such bandwidths are becoming insufficient to facilitate communication and the presentation of complex data mainly because of the bandwidth requirements of images and video.

The recent emergence of Asynchronous Transfer Mode technology, however, provides a unique opportunity to become a *driving technology* for *high performance computing platforms*, with line speeds that will scale to interfaces in the gigabit range. Unlike shared-medium LAN technologies in which users compete for bandwidth, the ATM network is capable of dynamically allocating bandwidth as a function of priority. The primary benefits that ATM technology offers over competing technologies are the ability of offering bandwidth-on-demand for a wide variety of applications (such as data, voice, video) with different latency and delay requirements, and the ability to support all this traffic with a guaranteed QoS.

ATM technology is recognized as the switching and multiplexing solution for Broadband Integrated Services Networks (B-ISDN) and is expected to be the basis for all future high-speed networks. ATM's efficiency can be attributed mainly to two facts: Making use of small fixed-size cells consisting of 5 bytes header and 48 bytes information field (payload), and dealing with virtual path/virtual channel connection. The absence of variable-length packets in an ATM network eliminates excessive delay variation because the small fixed-size cell enables extremely high switching speeds. Establishing virtual connection by using the address fields in the 5-byte ATM cell header, ATM separates the traffic from the physical channel and allows many users to share links. Over a single virtual path, many individual application streams can be multiplexed together, by allowing the full use of it's available bandwidth. Fig. 1 gives the scenarios of VPCs / VCCs multiplexing and VP / VC switching.

ATM technology is able to carry different kinds of traffic over the network with a guaranteed QoS. To achieve this QoS, a *traffic contract* is secured between the application and the network in which the application informs the network of its anticipated traffic characteristics such as a peak and an average data rate, a maximum delay, and a cell loss probability of each direction of the requested connection upon an initial set-up. The network then takes these traffic parameters and available resources into account in allocating its resources to satisfy the application's requests without adversely affecting existing connections. However, *congestion* happens when multiple cells arriving simultaneously through different incoming links attempt to reach the same outgoing link during the same cell slot time. This is because, unlike the deterministic multiplexing where each connection is allocated its peak bandwidth, statistical multiplexing allows several connections which might be very bursty at times, to share the same link based on their traffic characteristics in the hope that statistically they will not all burst, at the same time. Thus, the flow rate is within the link bandwidth most of the time. In other words, it allows the sum of the peak bandwidth requirement of all connections on a link to even exceed the aggregate available bandwidth of the link under some condition of discipline. In this context, a certain degree of "overbooking" is potential to degrade the QoS, therefore congestion control is particularly important to providing guaranteed QoS for uses in the ATM network.

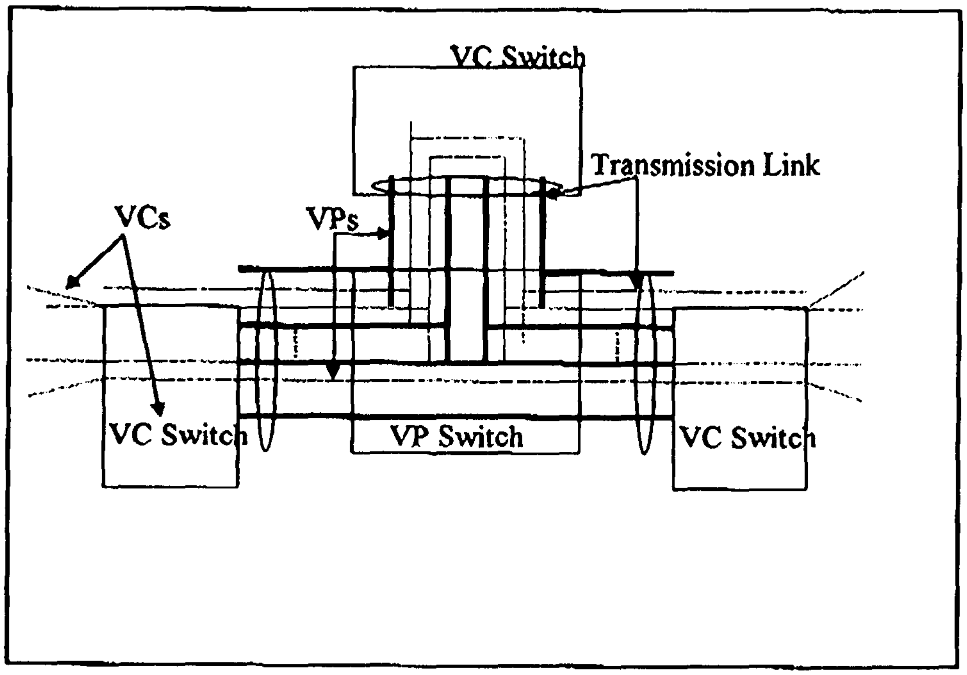

Fig. 1. VPC/VCC Multiplexing and Switching

There are a number of mechanisms (both preventive and reactive) to avoid congestion in ATM network: for example, Usage Parameter Control (UPC)/Network Parameter Control (NPC), Connection Admission Control (CAC), rate-based control, and so forth. However, a *congestion control through adequate buffering*, is becoming particularly significant to minimize the probability of cell loss and cell delay. In this case, only one cell is allowed to go through the network while the others must be stored in buffers. At this time, a switch buffering strategy as well as the buffer size become important because buffers are required to secure low cell-loss rate by providing a place to guard against cell loss when the switch is overloaded with *bursty traffic*. The choice of either of them can have a dramatic effect on the performance of the switch. If cell loss is experienced due to overflowing buffers, this will introduce a degradation in the overall system performance. The goal of this paper is to demonstrate a new design of a threshold based dynamic buffer allocating switch using OPNET simulation package from Mil.3, Inc.[3].

2. DYNAMIC BUFFER MANAGEMENT

Under bursty traffic, statistical multiplexing of ATM cells will lead to severe cell loss. Huge amount of buffers cause excessive cell delay and switch cost, therefore an appropriate number of buffers are allocated in a switch and we seek an optimal utilization of them. The goal is to ensure fair share of the buffer space and achieve low cell loss even under bursty loading conditions. To achieve this, a feasible solution is to use a dynamic buffering strategy which applies a *threshold based sharing policy* to the buffers and a *priority based cell pushed-out policy* to either buffered cells or incoming cells.

There are a number of buffer sharing schemes available through research in the past years, namely, (a) *complete sharing* where all cells are accommodated if the storage space is not full, (b) *complete partitioning* where the entire space is divided permanently to all output ports (in fact, it has the same performance as output queuing does), (c) *partial sharing/partial partitioning* where a number of buffers are always reserved while the rest are partitioned among the output ports and so on [4,5,6,7]. Intuitively, the threshold (for a logical output queue) itself should be dynamically changing rather than statically partitioning for the sake of adapting the different mixes of incoming traffic [8,9,10]. A scheme of complete sharing based on virtual partition will achieve better performance than complete partition [11]. However, the statistical nature of the significant part of the traffic, its burstiness and variability combine to pose difficulties in deciding the appropriate partition of the fixed size buffers. Therefore, in a statistical multiplexing environment with heterogeneous traffic, before ending up with an optimal threshold, it is quite important to define a *congestion detection criteria* for adaptive threshold updating. The complexity and the performance of the switch mechanism play a critical role in such a preventive congestion control. Therefore, to detect the congestion and update the corresponding threshold in real time, several traffic attributes on each outgoing link are taken into account in our algorithm. There are a number of factors which help gauge the traffic volume.

The first two factors, *in-use bandwidth* and *in-use bandwidth of distinct QoS of an incoming traffic*, indicate the anticipated volume as well as the QoS requirement of the incoming traffic on each outgoing link in general. The last two factors, *available number of buffer on the corresponding logical queue* and *the latest short-term cell arrival rate destined to the corresponding output port*, reveal the real-time evolution state of the switch. This information gives a heuristic estimation of the traffic in the near future. Therefore, threshold updating decision will be made based on this estimation.

The ATM differentiates various kinds of QoS for the traffic in terms of the bounds on the bandwidth and the cell loss ratio. There is a serious need to incorporate explicit burstiness modeling in the traffic models used for performance analysis. It has been found that the traffic parameters, such as inter-arrival time and loss probabilities are very sensitive to the assumed source characteristic. To simulate the traffic types in general, we would like to categorize them into two classes: real-time traffic, which includes interactive video, voice and the like, and nonreal-time traffic, such as data. We assume all real-time cells irrespective of their connections tolerate the same delay and cell loss ratio, and the nonreal-time traffic can bear more delay and higher cell loss ratio than real-time traffic. Nevertheless, we make another assumption that all traffic coming to the switch conforms to its negotiated traffic contract, therefore there is no malicious as well as unintentional user misbehavior involved in the congestion condition. In other words, the congestion happens because of the statistical nature of the traffic of ATM network. Thus, we will show high throughput over an ATM network will come from the dynamic buffering strategy presented in this paper.

3. OPNET SIMULATION

The proposed switch has a complete shared buffer architecture utilizing a threshold based virtual partitioning strategy of the buffer space. In order to evaluate its performance, simulation models of ATM network containing this kind of switch are used in this study. They are built through OPtimized Network Engineering Tools (OPNET). OPNET of Mil3, Inc, provides a very sophisticated software package capable of supporting simulation and performance evaluation of communication networks and distributed systems [3].

To build a model representing a real-world network, OPNET allows a model specification, complete with network, node, process and parameter editors to capture the characteristics of a modeled system's behavior at different modeling hierarchies. Geographically distributed sites are referred to as nodes or sub-networks. These nodes are connected through point-to-point links, bus links, or radio links in the OPNET network editor. A sub-network consists of a group of nodes along with the links connecting them within a LAN. A node is a facility (for example, a computer) or resource in which data either originates, or is received and processed, or both. The OPNET node editor allows user to classify a node's internal behavior into a variety of different modules. These modules are data creation, data transmission, data storage, data routing, and so forth. Modules are linked by packet stream or statistic wires. Normally, there is no limitation on the number of modules within a node. The Process editor further defines a module in a finite state machine (FSM) fashion. A FSM consists of states and transition conditions. It depicts the overall logical organization of the module. In this level, an FSM diagram is a run-able unit called a process, which can in many ways be thought of as similar to a traditional software program. The operations performed in each state or on transition conditions are written in a high level language called Proc-C. Proc-C has extensive support for communications and distributed systems, contains library of over 275 kernel procedures, and allows realistic modeling of all communications protocols and algorithms. Finally, the Parameter editor is capable of defining the format of data transmitted among modules, between modules and simulation kernel, etc. For example, a 53-bytes ATM cell is specified by this tool.

A simulation is executed by taking the simulation program and a set of data files representing the model's parameters to dynamically model the behavior of the actual system. Pre-defined statistics of interest can be collected on sample simulation runs by using the Probe Editor to specify which built-in statistics should be recorded by the Simulation Kernel. Each simulation run can be viewed as an experiment on a certain group of parameters of the system. It exhibits a repeatable behavior each time without changing the execution environment. This property is useful for isolating problems and analyzing or demonstrating interesting behavior. However, for simulating stochastic elements (for example, a packet generator module) different seed (one of the parameters) values should be used to produce distinct sequences so that each particular simulation run can then be thought of as representing one possible scenario of events for the modeled system. Statistics of interest can be collected as output vector files during the simulation runs.

Fig. 2 illustrates a typical ATM node model used in our simulation study, which can serve as an ingredient of an upper level sub-net of an ATM based network. A node shown in Fig. 2 usually serves as a source node in an ATM network. The ATM adaptation layer (AAL) module provides AAL signaling and AAL5 data transfer services. Its functionality is carried out by three distinct processes: *ams_aal_disp* process for incoming signal dispatching; *ams_saal* process for call establishment and release signaling; and *ams_aal5_conn* process for data transfer for AAL5 connections [3].

The four modules in the central square that perform the ATM functions are: control message managing, relay cell conformance checking/relay cell delivering (within the node), ATM layer interfacing, and fast cell switching and are carried out by ATM Management, ATM Translation, ATM Layer and ATM switch process respectively.

The *ATM Management module* processes all ATM control messages including setting up and releasing connection messages, route updating messages, and so forth. Based upon these messages, calls originating from and arriving at the current node, are accepted by allocating a portion of their peak bandwidth, in order to obtain a high statistical multiplexing gain. The Static Distributed Routing (SDR) scheme used in this module searches for the next node through the best known route. The *ATM Translation module* identifies cells into data and control cells. Data cells are either forwarded to the ATM layer module, if they are destined for the current node, or assigned new VPI/VCI values and sent to the ATM Switch module. All control cells are forwarded to the ATM Management module. Since we assume congestion only happens due to the statistical nature of the incoming traffic, any cell going through this module will be regarded as a legal cell which conforms to its traffic contract. Therefore, Usage Parameter Control (UPC) function is not switched on here. The *ATM Layer module* acts as the gateway between AAL and the ATM function layer. It packages outgoing information from higher layer (via AAL) or from the ATM Management module to ATM Switch module in the form of ATM cells. It also extracts information from incoming ATM cells to the AAL module. The *ATM Switch module* is a fast packet switch for all outgoing ATM cells. It en-queues the cells in a shared buffer pool and sends them to the transmitter by waiting for the next cell slot time. To guard cells against cell loss, effort should be spent on making full use of buffers installed here.

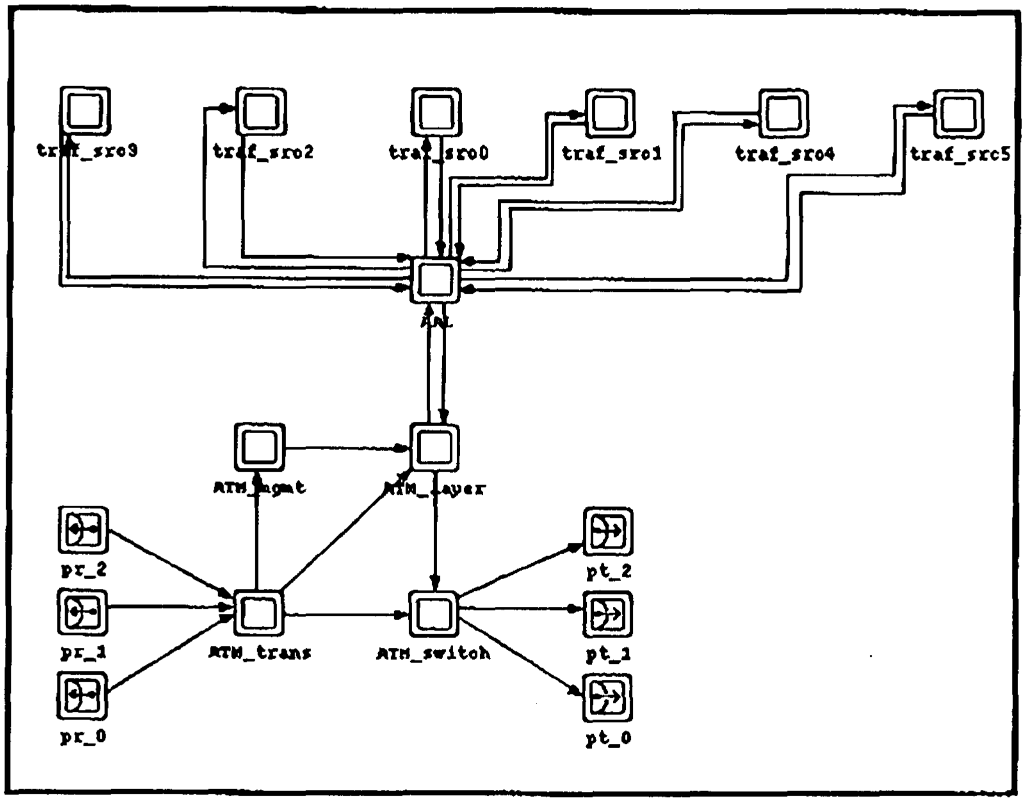

Fig. 2. A typical source node model in an ATM network.

Outside the central square, there are three other kinds of modules. The point-to-point receiver/transmitter (pr/pt) pair can be viewed as the end point of a point-to-point link in the top most hierarchy (in network editor). The pair is the data entrance/exit from/to other nodes.

The *Traffic Source module* generates the actual packets and sends them to the AAL module. Here, an on-off type of traffic is chosen to study. Traffic is characterized by call duration time, call wait time, mean packet inter-arrival time, packet size, and QoS. The active state of the traffic, in which cells are generated at peak rate, is interleaved with idle state, in which no cell comes out from the module. The active and idle states are exponentially distributed. By replacing the Traffic Source module with the Traffic Sink module and deleting two pairs of point-to-point receiver/transmitter, a sink node is constructed shown in Fig. 3. The Traffic Sink module acts as the final destination for the data packets.

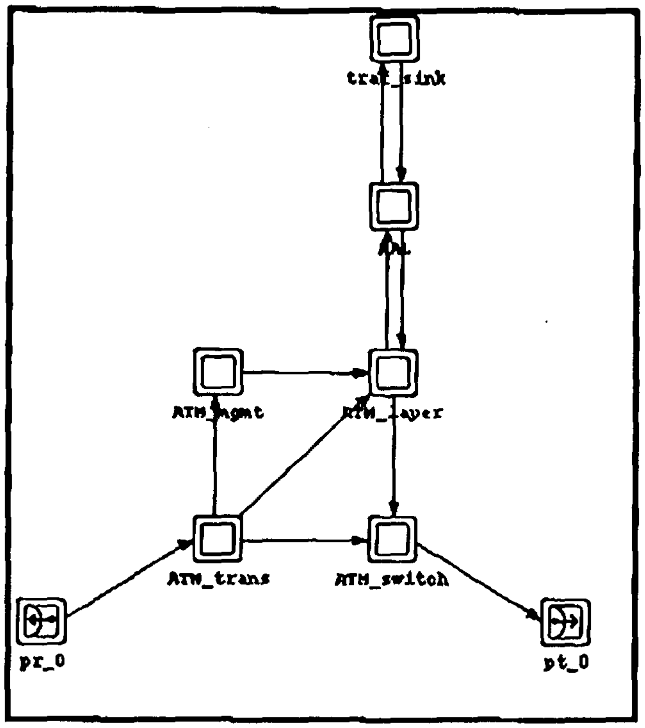

Fig. 3. A typical sink node model in ATM network

4. DYNAMIC BUFFERING ALGORITHM IN THE ATM SWITCH MODULE

The static management of the buffer space has poor performance when traffic headed to one output port is heavy while the rest is very light. The most significant improvement of the static buffer management is to eliminate the static way of sharing the buffers as much as possible. Therefore, our solution to accommodate more cells of the traffic, consists of dynamically sharing the limited buffer space among all output ports while maintaining a dynamic threshold on each logical output port queue, and applying cell-accommodation-rule (CAR) to each incoming cell.

Two additional variables were created to be used in the algorithm. A *buffer_pool* variable of *AtmT_Switch_Buffer_Desc* type of new process state variable holds the maximum number of available buffers for this particular switch, and indicates the current buffer occupancy. A *port_queue* variable of *AtmT_Switch_Port_Queue_Desc* records the newest threshold of each output port associated with the switch.

There are six general assumptions in the proposed algorithm which use the above newly defined variables:

1. First In First Out (FIFO) policy is enforced at all times.
2. Real-time and nonreal-time cells have equal chances to be transmitted over an outgoing link from the corresponding output port.
3. Thresholds of the logical queues for all output ports are equally assigned in the initialization phase.
4. Every M time slots, thresholds are updated.
5. Cell-Accommodation-Rule is applied whenever a cell arrives at the switch.
6. Any excess of buffer space from any logical queue goes to the shared buffer pool.

With the assumptions 4 and 5, this algorithm is able to dynamically adapt the traffic and keep fair cell loss among all output ports. This algorithm differs from the static algorithm, in that cell buffering depends on thresholds of output ports rather than fixed length of each output queue; any buffer space in the buffer pool is potentially going to be occupied by cells destined to various outgoing links. Therefore, there is no dedicated use of buffer space any more.

To measure and predict the incoming traffic, we need to keep track of the following four factors on any outgoing link *i* (assume the total number of links is n):

1. In-use bandwidth.
2. In-use bandwidth of the real-time traffic.
3. Available buffer space of the corresponding logical queue.
4. The latest short-term cell arrival rate destined to corresponding output port.

The updating processes for thresholds are efficient in the sense that the four factors can all be easily retrieved, and it doesn't involve complicated calculation. Therefore it causes little latency over the network. Fig. 4 shows a portion of a C-code to compute the new threshold.

In addition to the adapting thresholds, a Cell-Accommodation-Rule (CAR) is needed to make use of the thresholds. Basically, when buffer pool is not full, any incoming cell should be accepted. When buffer pool are all filled in, however, a selective admission and push-out mechanism should be enforced.

If the length of the logical queue, where the cell headed to, is less than its threshold, it is obvious that there are some other queues whose length exceeds their thresholds. We denote this kind of queues as Q_{nc} (non-conforming queue) and the remainder of queues as Q_c (conforming queue). In this case, a new cell is buffered by dropping one buffered cell from either Q_{nc} or Q_c. The *selection policy* is listed as follows with descending priority:

1. A real-time cell from either Q_{nc} or Q_c whose Cell-Delay-Variation (CDV) exceeds the tolerable limit in any kind of queue.
2. A nonreal-time cell from Q_{nc} queue where its position is outside of the threshold of its corresponding queue.
3. A nonreal-time cell in the same queue where the new arrival is destined.
4. A real-time cell from Q_{nc} queue where its position is outside of the threshold of its corresponding queue.

If the logical queue where the cell destined has its length equal to/greater than its threshold, then the cell is either *accommodated or dropped* depending on the following situation:

5. If there is a real-time cell in Q_{nc} or Q_c whose CDV exceeds the tolerable limit, then it gets dropped. Else,
6. If there are nonreal-time cells in the same queue, then the newest nonreal-time cell is dropped. Else,
7. If the new arrival is a nonreal-time cell, it gets dropped immediately. Else,
8. If the new arrival is a real-time cell, it either is dropped immediately, if no nonreal-time cell exists in all queues, or replaces a newest nonreal-time cell in some other queue.

```
/*compute the new threshold*/
                  if (X==0)
                        {
                        /* Handle the exception case 1 of no call being routed
                        through the link */ .
                        for (port_index = 0; port_index < port_count;
                              port_index++)
                        {
                        /* Obtain the i-th port descriptor pointer. */
                        pdesc_ptr = &pdata_ptr->port_array [port_index];

                        /* Check whether the port_index is the last one, if
                        notassign the threshold = buffer_pool.max/port_count,
                        otherwise threshold = the rest of buffer space, it
                        might not be as much as others*/
```

```
                if (port_index != port_count - 1)
                      port_queue [port_index).threshold =
                            buffer_pool.max_bufsize/port_count;
                else
                      port_queue [port_index).threshold =
                            buffer_pool.max_bufsize -
                            (buffer_pool.max_bufsize/port_count)*
                            (port_count-1);

                /* set the other field of port_queue[port_index] */
                port_queue [port_index).latest_arrival = 0;
                }
                } /*End of exception case 1 */
          else  {
                bufspace = buffer_pool.max_bufsize;
                for (port_index = 0; port_index < port_count;
                      port_index++)
                {
                /* Check whether the port_index is the last one*/
                if (port_index != port_count -1)
                      {
                if (x[port_index]== 0)
                            port_queue [port_index).threshold =
                            floor(bufspace*DEFAULT_SHARE);
                      else
                            {
                            if (x[port_index]/X == 1)
                                  port_queue [port_index).threshold =
                            floor(bufspace*(1-DEFAULT_SHARE));
                            else
                                  port_queue [port_index].threshold =
                                  floor(bufspace*x[port_index]/X);
                            }

                      bufspace -= port_queue (port_index).threshold;
                      }
                else
                      port_queue [port_index].threshold = bufspace;

                /* set the other field of port_queue[port_index) */
                port_queue [port_index).latest_arrival = 0;
                } /*End for */

                }
```

Fig. 4. C-code to compute the new threshold.

5. SIMULATION STUDY

Fig. 5 shows the topmost layer of an ATM network model constructed by integrating one source node (f_0) and three sink nodes (f_1, f_2, f_3). Each sink node has its own unique ATM network address as its node serial number in the network. Traffic source generates either real-time or nonreal-time traffic destined to one sink node. The link rate is set to 51.84 Mbits/sec, which corresponds to *STS-1*. The Segmentation and Assembly Rate (SAR) has been kept very high so that there is no traffic shaping due to the AAL5 layer. Since packet size is fixed as 320 bits, segmentation is not required.

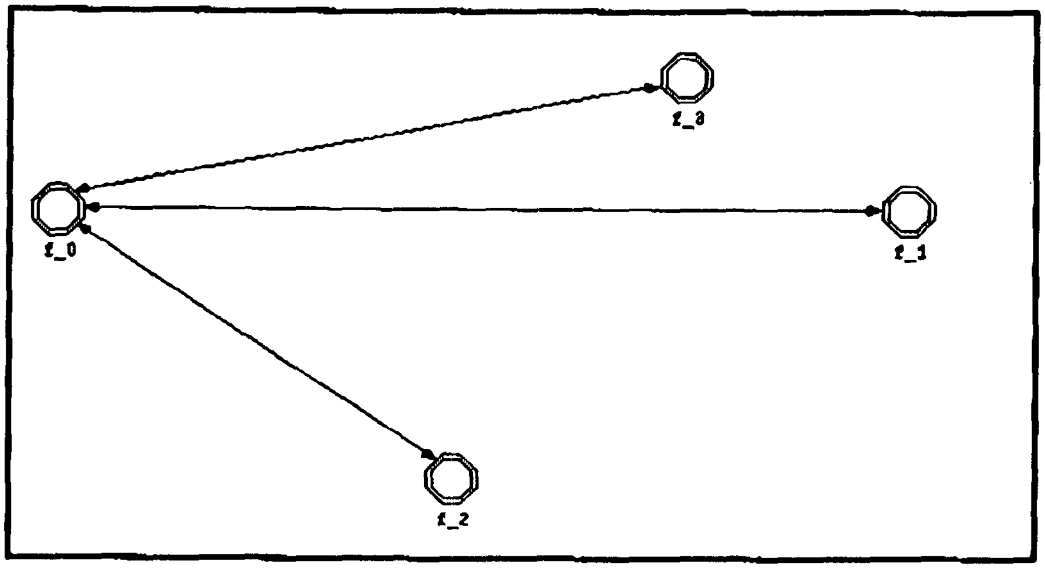

Fig. 5. An ATM network model used for this study.

Two distinct models were created by using a static complete partition based ATM Switch process in one model, and a dynamic shared buffer based ATM switch process in the other one. Switch performance based on static strategy versus dynamic strategy are then evaluated.

Table 1 depicts a typical traffic parameters used in our simulation. Other than a packet size, traffic from an individual application is characterized also by call duration time, call wait time, mean packet inter-arrival time, QoS and destination. Table 2 shows the buffer allocation in each buffering strategy. The total buffer size for the dynamic buffering strategy switch simulation is 5022 cells. For the static buffering strategy, 5022-cell buffer space is equally assigned to 3 output port queues, and 950-cell space out of 1674-cell space is used for accommodating real-time traffic in each output port. The remainder is utilized for nonreal-time traffic.

Table 1. Traffic parameter for investigating the performance of static buffering strategy versus dynamic buffering strategy.

	traf_src0	traf_src1	traf_src2	traf_src3	traf_src4	traf_src5
mean peak interarrival time	0.00009	0.00001	0.00001	0.000012	0.000012	0.00001
call duration time	0.23	0.2	0.2	0.2	0.156	0.2
call wait time	0.2	0.415	0.51	0.4	0.5	0.32
QoS	nonreal-time	real-time	nonreal-time	real-time	real-time	nonreal-time
destination	1	2	3	1	3	2

Since the packet interarrival time is exponentially distributed, by selecting different seed in our simulations, the mixed pattern of a bursty traffic is obtained. Simulation results in terms of cell loss are shown in Fig. 6 and Fig. 7. Fig. 6 illustrates the cell loss in dynamic buffing strategy switch, where the threshold updating frequency M = 500, cell loss happened at output port 1. The nonreal-time cell loss began on the 0.5244 second. When the simulation stopped, the total cell loss is 2212 cells. There were no real-time cells dropped during this period of time. On the other hand, Fig. 7 shows the cell loss in dynamic buffering strategy at frequency M=3000.

Table 2. Buffer allocation in two distinct simulations.

	real-time traffic buffer size	nonreal-time traffic buffer size
output port 0, 1, 2 (static buffering)	950	724
buffer pool (dynamic buffering)	5022	

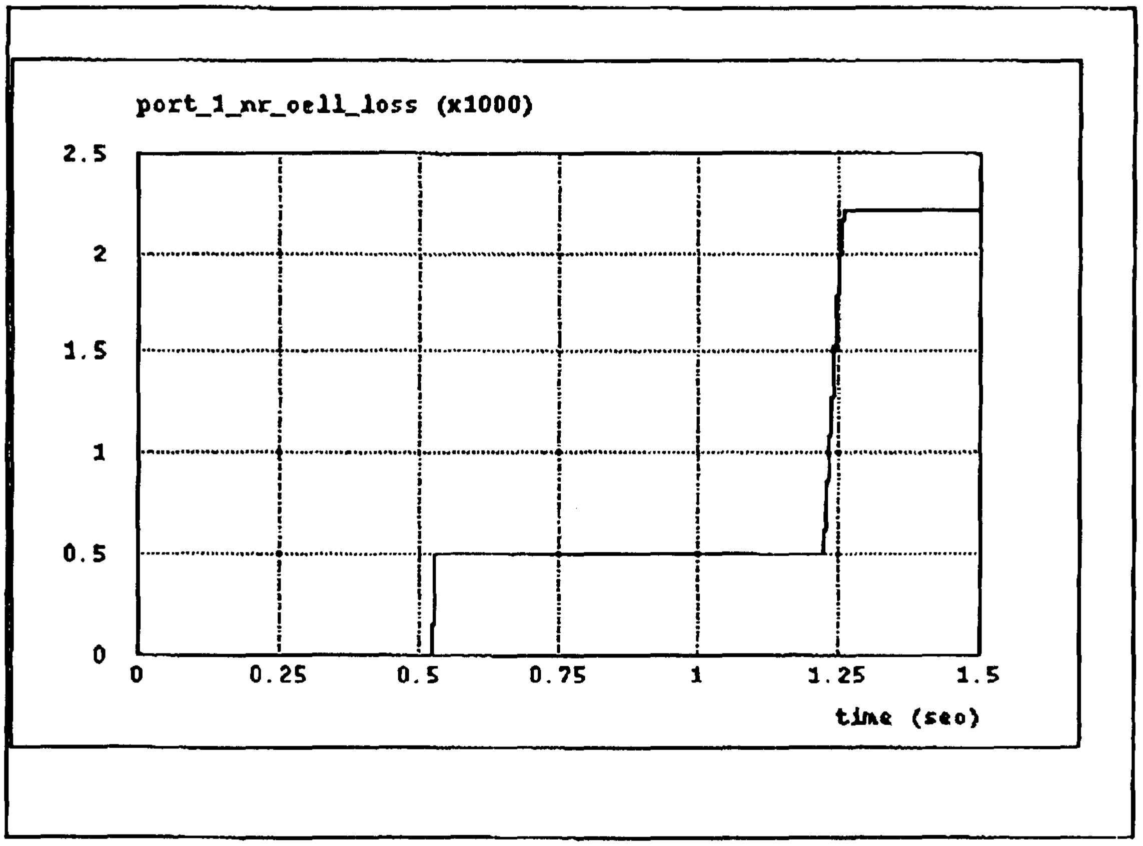

Fig. 6. Cell loss in a dynamic buffering strategy switch.

6. CONCLUSION

A new module for use in the OPNET simulation package was written to reallocate the buffer space dynamically. The Proc-C code and the simulation results of this study were shown. Advantages of the threshold based dynamic buffering strategy have been examined. For the purpose of getting efficient and fair use of buffer space in ATM switches, we have simulated a threshold based dynamic buffer allocation scheme in the ATM switch, and then the system behavior under varying on-off bursty traffic pattern is investigated via simulation.

Under a heavy traffic load, the frequency of the threshold updating plays a critical role to gain higher throughput and to maintain fair cell loss among all output ports.

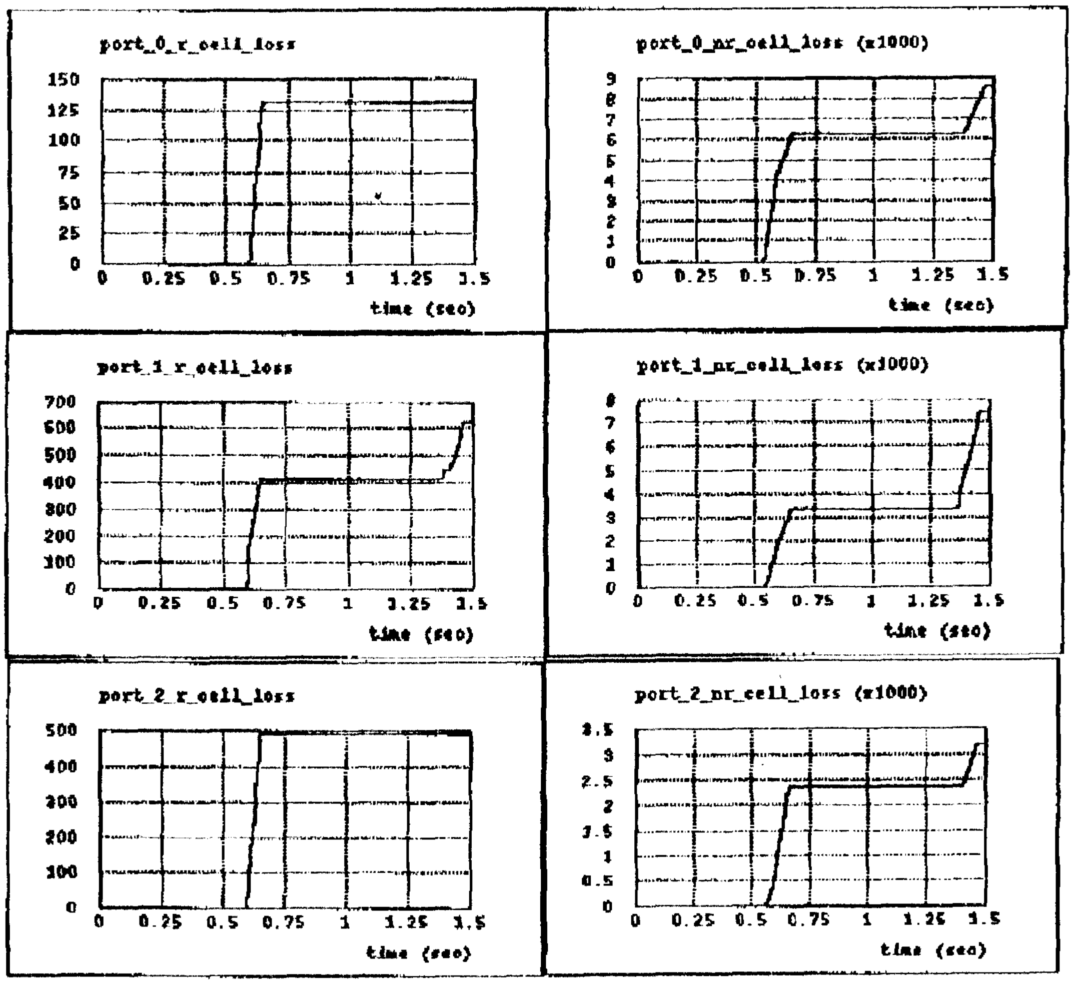

Fig. 7. Cell loss under threshold updating frequency M = 3000 slots

ACKNOWLEDGEMENTS

This work has been supported in part by Jet Propulsion Laboratory, California Institute of Technology under Contract No. 960414. Dr. Chronopoulos acknowledges the NSF grant ASC-9634775, which provided partial support for this research.

BIBLIOGRAPHY

[1] David E. McDysan and Darren Spohn, 'ATM Theory and Application', McGraw-Hill, Inc., 1994.

[2] William Stallings, 'Data and Computer Communicatations', MacMillan Publishing, Co., 1994.

[3] Mil 3, Inc., "ATM Model Description", OPNET Example Models Manual, Chapter ATM.

[4] Leandros Tassiulas, Yao Chung Hung, and Shivendra S. Panwar , "Optimal Buffer Control During Congestion in an ATM Network Node, " *IEEE/ACM Transactions on Networking*, vol. 2, no. 4, pp.374-386, Aug. 1994.

[5] W. Kowalk and R. Lehnert, "The Policing Function to Control User Access in ATM Networks - Definition and Implementation, IEEE ISSLS, '88, Boston, MA, pp.240-245, Sept. 1988.

[6] Israel Cidon, Leonidas Georgiadis, Roch Gue'rin, and Asad Khamisy, "Optimal Buffer Sharing", IEEE Journal on Selected Areas in Communications, Vol. 13. No. 7, September, 1995.

[7] Gerd Niestegge, "The Leaky Bucket Policing Method in the ATM Network," *International Journal of Digital and Analog Communication Systems*, vol.3, no.2, pp.187-197, June 1990.

[8] Mark J. Karol, Michael G. Hluchyj and Samuel P. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch", IEEE Transactions on Communications, Vol. COM-35, No. 12, December, 1987.

[9] Anwar Elwalid, Debasis Mitra, and Robert H. Wentworth, "A New Approach for Allocating Buffers and Bandwidth to Heterogeneous, Regulated Traffic in an ATM Node", IEEE Journal on Selected Areas in Communications, Vol. 13. No. 6, August 1995.

[10] A. Iwata, N. Mori, C. Ikeda, H. Suzuki, and M. Ott, "ATM connection and Traffic Management Schemes for Multimedia Internetworking", Communication of the ACM, February 1995/Vol.38.No.2

[11] Guo-Liang Wu, Jon W. Mark, "A Buffer Allocation Scheme for ATM Networks: Complete Sharing Based on Virtual Partition", IEEE Transaction on Networking. Vol. 3. No. 6. December 1995